Outline

- .NET Framework
- Object Oriented Programming (OOP)
- C# basics
- Database programming using ADO.NET
.NET Framework

- Development Languages
  - C++, C#, J#, Python, Perl, ...

- Integrated Development Environment (IDEs)
  - Visual Studio .NET, Mono Project

- Common Language Infrastructure (CLI)
Common Language Infrastructure (CLI)

- **Microsoft's** implementation of the CLI is called the Common Language Runtime (CLR)

CLR primary components:
- Common Type System (CTS)
  - Type definitions independent of any specific programming language
- Common Language Specification (CLS)
- Just-In-Time compiler (JIT) and Intermediate Language (IL)
  - Language Independent
  - Abstraction similar to Java’s Virtual Machine
- Virtual Execution System (VES)
  - Memory management and garbage collection
  - Exception handling
.NET Framework (Cont’d)

- Documentation, examples, tips:
  - Microsoft Developer Network (MSDN)
  - Search Engines
    - Google code search (http://www.google.com/codesearch)
  - Programming websites and forums
    - Most of the times your question has been answered before – needs patience and good search keywords!!
  - Books, online tutorials, ...
OOP: A software construction paradigm

- **Objects**
  - Abstract concepts, basic entities that describe internal information (variables) and the means to access it, change its state, perform actions (functions)
  - In C# (like Java and most OOP languages) everything is an object

- **Classes**
  - Programmer-defined types that model the parts of the system (objects)
  - Basically a blueprint

- **Instances**
  - Building blocks of the code
  - Allocated in memory sharing the code but not the data

- **Methods/Variables**
  - Primary mechanisms (code common to all instances of a class) and data variables
  - Static parts: belong to a class, accessible and shared by all instances
OOP (Cont’d)

Every object has its own penNumber.

Sheep class acts as a template for creating sheep objects.

Instance “dave” created from Sheep class.

The class keeps track of totalSheep.
OOP (Cont’d)

- Abstraction
- Inheritance
- Polymorphism
- Encapsulation

Simple example:

```java
public class MyNumber {
    int val = 20;

    public int value() {
        return val;
    }

    public int increase(int how_much) {
        val += how_much; // val = val + how much;
        return val;
    }
}

MyNumber m = new MyNumber();
int l = m.value(); // l will be 20
int k = m.increase(12); // k will be 32
```
C# Basics

- Case sensitive
- Common data types (primitives):
  - bool, byte, short/int/long, float/double, char/string
- Conversion
  - implicitly
    - float f = 56;
  - by casting
    - int i = (int)5.6;
  - with convert class
    - int i = System.Convert.ToInt32(3.25d);
  - provided methods
    - string s = (5.25).ToString();
Type Casting

- **Int ⇔ byte array (4 bytes)**
  - `byte [] b = BitConverter.GetBytes(integer);`
  - `int i = BitConverter.ToInt32(b, 0);`

- **float ⇔ byte array (4 bytes)**
  - `byte[] b = BitConverter.GetBytes(float);`
  - `float f = BitConverter.ToSingle(b, 0);`

- **String ⇔ byte array**
  - `byte[] b = new ASCIIEncoding().GetBytes(string);`
  - `string s = new ASCIIEncoding().GetString(b, 0, length);`
C# Basics II

- Operators:
  - +, -, *, /, %
  - &, |, ..., &&, ||, ...
  - ++, --...
  - <, >, ==, ...
- Decision making:
  - if...then...else
  - do...while, while, for
  - Foreach
    - e.g. foreach (int i in list_of_ints) { ... }
  - switch
- Declarations and scope
- Comments like c++ (try Ctrl-K-C and Ctrl-K-U for mass commenting ;-)
C# Classes

- Everything is a class derived from class Object
- Namespaces to avoid confusion
- Create new instances with keyword `new`
  ```csharp
  string s1 = new string('T', 3); // s1 = “TTT”
  ```
- Reference their methods with dot (.)
  ```csharp
  string s2 = s2.ToLower(); // s2 = “ttt”
  ```
- Public/private, static, final
- Try looking at existing examples all around if unfamiliar with syntax; may find it easier to edit existing code at first
C#: Hello world!

```csharp
using System;

class MyApp
{
    static void Main()
    {
        Console.WriteLine("Hello world!");
    }
}
```

- To execute this you need:
  - Install .NET Framework (you already have if you installed VS.NET)
  - Write the code as text in a file, e.g. MyApp.cs
  - Compile it with C# compiler (csc.exe)
    - `%Windows_Directory%\Microsoft.NET\Framework\v1.1.4322\csc.exe`
    - Add it to path for convenience
  - Run the produced executable e.g. MyApp.exe
C#: Useful libraries

- System.Math class
e.g. `double d = Math.Pow(5,3); // d = 5^3`

- System.Collections class
e.g. ArrayList, Hashtable, Stack, SortedList, etc.
  ```csharp
  ArrayList a = new ArrayList();
a.Add(5); a.Add(6.7);
  int i = (int)a[0];
  double d = (double)a[1];
  ```

- Regular expressions
  (System.Text.RegularExpressions)

- File Input/Output (System.IO)

C#: Console

- Two very common functions for console applications:
  - `Write(...)`: Writes the specified information to the standard output stream
  - `WriteLine(...)`: Like `Write()` only followed by the current line terminator
  - `Read(...)`: Reads the next character from the standard input stream
  - `ReadLine(...)`: Reads the next line of characters from the standard input stream

- To prevent the execution of a program from terminating when done you can add all the code in a try-block with a `Console.ReadLine()` in the finally-block (see next slide for exception handling)
C#: Exceptions

```csharp
try {
    ...
    //code that can generate errors
    //and throw exceptions
}
catch (ExceptionType1 ex1) {
    ...
    //code to handle type1 exceptions
}
catch (ExceptionType2 ex2) {
    ...
    //code to handle type2 exceptions
}
catch {
    ...
    //code to handle the rest of the exceptions
}
finally {
    ...
    //code to be executed regardless
}
```
Visual Studio.NET

- All sorts of projects (Windows apps, ASP projects, Web Services, etc)

- New C# project:
  - File → New → Project
  - Choose “Visual C# Projects” and
    - Console Application for simple console programs
    - Windows Application for a project with GUI
  - Choose project’s name and path

- Possibly the project will contain VS.NET generated code
Database Connectivity

- Microsoft Data Access Components (MDAC)
  - ActiveX Data Objects (ADO)
  - Object Linking and Embedding Database
- Open Database Connectivity (ODBC)

*Note: the Microsoft SQL Server Network Library (Net-Lib) is used specifically by SQL Server but is still counted as an official part of MDAC.
ADO.NET

- High level programming interface to OLE DB
ADO.NET:
Managed Provider Classes

- Three different sets of classes for accessing:
  - SQL Server databases (System.Data.SqlClient)
  - OLE DB (System.Data.OleDb)
  - ODBC databases (System.Data.Odbc)
- Includes following classes:
  - Connection
    - Provides a connection used to communicate with the data source.
  - Command
    - Used to perform some action on the data source, such as reading, updating, or deleting relational data.
  - Parameter
    - Describes a single parameter to a command. A common example is a parameter to a stored procedure.
  - DataReader
    - A class used to efficiently process a large list of results one record at a time.
  - DataAdapter
    - A bridge used to transfer data between a data source and a DataSet object.
ADO.NET:
A simple example

using System;
using System.Data;
using System.Data.OleDb;

class SimpleExample {
    public static void Main() {
        ...
    }
}
}
ADO.NET: OleDbConnection

```csharp
string connString = "Provider=Microsoft.Jet.OLEDB.4.0;
    Data Source=C:\\Test Files\\Sample.mdb;
    User Id=admin;Password=;"
OleDbConnection conn = new OleDbConnection(connString);
...
conn.Open();
...
conn.Close();
```

- Connection string specifies server, database, user & password (if needed)
- Closed connections are added to a pool to be reused if the same connection string is used. Saves much time in comparison to establishing a new connection

- Public Properties
  - ConnectionString, ConnectionTimeout, Database, DatabaseSource, ServerVersion, State, WorkstationId

- Public Methods
  - BeginTransaction(), ChangeDatabase(), Close(), CreateCommand(), Open()
ADO.NET: OleDb Command

OleDbCommand cmd = new OleDbCommand();
    cmd.Connection = conn;
    cmd.CommandText = "DELETE FROM Students";

Or:
OleDbCommand cmd =
    new OleDbCommand("DELETE FROM Students",conn);

- One can also be created by executing CreateCommand() on a Connection instance (matter of taste)
- Needs an open connection before execution

- Public Properties
  - CommandText, CommandTimeout, CommandType, Connection, DesignTimeVisible, Parameters, Transaction, UpdatedRowSource

- Public Methods
  - Cancel(), CreateParameter(), ExecuteNonQuery(), ExecuteReader(), ExecuteScalar(), ExecuteXmlReader(), Prepare(), ResetCommandTimeout()
ADO.NET: ExecuteNonQuery()

```csharp
cmd.ExecuteNonQuery()
```

or

```csharp
int rowsAffected = cmd.ExecuteNonQuery();
```

- Returns the number of database rows affected by the command, if any.
- Can be used for
  - SQL INSERT, UPDATE, and DELETE statements, stored procedures that don't return a value
  - Data Definition Language (DDL) statements such as CREATE TABLE and CREATE INDEX
ADO.NET: ExecuteReader()

OleDbDataReader myReader = mySqlCommand.ExecuteReader();

- Requires an open connection
- You need to close the reader as soon as you’re done to release the resources

- The execution is parametrized. Possible values: CloseConnection, Default, KeyInfo, SchemaOnly, SequentialAccess, SingleResult, SingleRow
  Usually issued without any (default)
ADO.NET: OleDbDataReader

while (myReader.Read()) {
    Console.WriteLine(myReader["FirstName"]);
    Console.WriteLine(myReader[0]);
}

Second way is faster but depends on schema or query

myReader[ myReader.GetOrdinal("FirstName") ]

IsDBNull() should be used to avoid an OleDbException

Public Properties
- Depth, FieldCount, IsClosed, RecordsAffected

Public Methods
- GetBoolean(), GetByte(), GetBytes(), GetChar(), GetChars(),
  GetDataTypeName(), GetDateTime(), GetDecimal(), GetDouble(),
  GetField(), GetFloat(), GetGuid(), GetInt16(), GetInt32(),
  GetInt64(), GetName(), GetOrdinal(), GetSchemaTable(),
  GetSqlBinary(), GetSqlBoolean(), GetSqlByte(), GetSqlDateTime(),
  GetSqlDecimal(), GetSqlDouble(), GetSqlGuid(), GetSqlInt16(),
  GetSqlInt32(), GetSqlInt64(), GetSqlMoney(), GetSqlSingle(),
  GetSqlString(), GetSqlValue(), GetSqlValues(), GetString(), GetValue(),
  GetValues(), IsDBNull(), NextResult(), Read()
ADO.NET:
OleDbDataReader (cont.)

OleDbCommand cmd = OleDb.Connection.CreateCommand();
cmd.CommandText = "SELECT * FROM Students;
SELECT * FROM Faculty;"
...
OleDbDataReader myReader = cmd.ExecuteReader();
...
do {
    while (myReader.Read()) {
        Console.WriteLine(..);
        ...
    }
} while (myReader.NextResult());
ADO.NET:
OleDbTransaction

OleDbTransaction trans = conn.BeginTransaction();

... cmd.Transaction = trans;

... cmd.ExecuteNonQuery();

... trans.Commit();
OR
trans.Rollback();

☐ Either all actions succeed and become persistent or (at least) one fails and the rest get canceled

☐ try {

... catch (Exception ex) {

    if (trans != null)
        trans.Rollback();
    Console.Out.WriteLine("Exception: " + ex.Message);
    Console.Out.WriteLine(ex.StackTrace);
}

}
ADO.NET: DataSet objects

- Generic classes, independent of database
- Make local copy, modify and push changes back to the database
- Before you populate a DataSet you first need a Connection, a Command, and a DataAdapter
- Used for random access to data (not forward-only) and disconnected work
There’s only one way to learn a programming language well: experience through trial-and-error

Some tips

- Spend some time on programming
- Try things you don’t know how to do yet
- Question the source of your errors
- Try to do it efficiently and elegantly
- Use all public sources
- The .NET Framework provides classes for most of the common tasks but they’re not always obvious. Five minutes of searching in the help files can save you from reinventing the wheel
CS Distinguished Lecture

☐ Title:

“Declarative Data Services: This Is Your Data on SOA”

☐ Dr. Michael Carey - BEA Systems, Inc.

☐ Thursday, October 11, 2007

☐ Time: 4 PM - 5:30 PM

☐ Location: SSL 150
References

- Wikipedia
- MSDN
- Google code search